IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Recognizer of Audio-Content in Digital Signals

Inventors:
M. Kivanç Mihçak
Ramarathnam Venkatesan

ATTORNEY's DOCKET NO. MS1-645US

TECHNICAL FIELD

This invention generally relates to a technology for improving the recognition of audio content of digital signals.

BACKGROUND

Digital audio signals offer many advantages over conventional media in terms of audio quality and ease of transmission. With the ever-increasing popularity of the Internet, digital audio clips have become a mainstay ingredient of the Web experience, buoyed by such advances as the increasing speed at which data is carried over the Internet and improvements in Internet multimedia technology for playing such audio clips. Everyday, numerous digital audio clips are added to Web sites around the world.

An audio "clip" indicates an audio signal (or bit stream), in whole or part. A clip may be stored and retrieved, transmitted and received, or the like.

As audio clip databases grow, the needs for indexing them and protecting copyrights in the audio clips are becoming increasingly important. The next generation of database management software will need to accommodate solutions for fast and efficient indexing of digital audio clips and protection of copyrights in those digital audio clips.

A hashing technique is one probable solution to the audio clip indexing and copyright protection problem. Hashing techniques are used in many areas such as database management, querying, cryptography, and many other fields involving large amounts of raw data. A hashing technique maps a large block of data (which may appear to be raw and unstructured) into relatively small and structured set of

identifiers (the identifiers are also referred to as "hash values" or simply "hash"). By introducing structure and order into raw data, the hashing technique drastically reduces the size of the raw data into short identifiers. It simplifies many data management issues and reduces the computational resources needed for accessing large databases.

Thus, one property of a good hashing technique is the ability to produce small-size hash values. Searching and sorting can be done much more efficiently on smaller identifiers as compared to the large raw data. For example, smaller identifiers can be more easily sorted and searched using standard methods. Thus, hashing generally yields greater benefits when smaller hash values are used.

Unfortunately, there is a point at which hash values become too small and begin to lose the desirable quality of uniquely representing a large mass of data items. That is, as the size of hash values decreases, it is increasingly likely that more than one distinct raw data can be mapped into the same hash value, an occurrence referred to as "collision". Mathematically, for an alphabet of cardinality A of each hash digit and a hash value length l, an upper bound of all possible hash values is A^l. If the number of distinct raw data is larger than this upper bound, collision will occur.

Accordingly, another property of a good hashing technique is to minimize the probability of collision. However, if considerable gain in the length of the hash values can be achieved, it is sometimes justified to tolerate collision. The length of the hash value is thus a trade off with probability of collision. A good hashing technique should minimize both the probability of collision and the length of the hash values. This is a concern for design of both hashing techniques in

compilers and message authentication codes (MACs) in cryptographic applications.

Good hashing techniques have long existed for many kinds of digital data. These functions have good characteristics and are well understood. The idea of a hashing technique for audio clip database management is very useful and potentially can be used in identifying audio clips for data retrieval and copyrights protection.

Unfortunately, while there are many good existing functions, digital audio clips present a unique set of challenges not experienced in other digital data, primarily due to the unique fact that audio clips are subject to evaluation by human listeners. A slight pitch or phase shifting of an audio clip does not make much difference to the human ear, but such changes appear very differently in the digital domain. Thus, when using conventional hashing functions, a shifted version of an audio clip generates a very different hash value as compared to that of the original audio clip, even though the audio clips sound essentially identical (i.e., perceptually same).

Another example is the deletion of a short block of time from an audio clip. If the deleted block is short and in an otherwise quiet portion of the clip, most people will not recognize this deletion in the audio clip itself, yet the digital data is altered significantly if viewed in the data domain.

Human ears are rather tolerant of certain changes in audio clips. For instance, human ears are less sensitive to changes in some ranges of frequency components of an audio clip than other ranges of frequency components. Human ears are also unable to catch small stretching and shrinking of short segments in audio clips.

Many of these characteristics of the human auditory system can be used advantageously in the delivery and presentation of digital audio clips. For instance, such characteristics enable compression schemes, like MP3, to compress audio clips with good results, even though some of the audio clip data may be lost or go unused. There are many audio clip restoration/enhancement algorithms available today that are specially tuned to the human auditory system. Commercial sound editing systems often include such algorithms.

At the same time, these characteristics of the human auditory system can be exploited for illegal or unscrupulous purposes. For example, a pirate may use advanced audio processing techniques to remove copyright notices or embedded watermarks from an audio clip without perceptually altering the audio clip. Such malicious changes to the audio clip are referred to as "attacks", and result in changes at the data domain.

Unfortunately, a human is unable to perceive these changes, allowing the pirate to successfully distribute unauthorized copies in an unlawful manner. Traditional hashing techniques are of little help because the original audio clip and pirated copy hash to very different hash values, even though the audio clips sound the same.

Common Attacks. The standard set of plausible attacks is itemized in the Request for Proposals (RFP) of IFPI (International Federation of the Phonographic Industry) and RIAA (Recording Industry Association of America). The RFP encapsulates the following security requirements:

- two successive D/A and A/D conversions,
- data reduction coding techniques such as MP3,
- adaptive transform coding (ATRAC),

- adaptive subband coding,
- Digital Audio Broadcasting (DAB),
- Dolby AC2 and AC3 systems,
- applying additive or multiplicative noise,
- applying a second Embedded Signal, using the same system, to a single program fragment,
- frequency response distortion corresponding to normal analogue frequency response controls such as bass, mid and treble controls, with maximum variation of 15 dB with respect to the original signal, and
- applying frequency notches with possible frequency hopping.

Accordingly, there is a need for a hashing technique for digital audio clips that allows slight changes to the audio clip which are tolerable or undetectable (i.e., imperceptible) to the human ear, yet do not result in a different hash value. For an audio clip hashing technique to be useful, it should accommodate the characteristics of the human auditory system and withstand various audio signal manipulation processes common to today's digital audio clip processing.

A good audio hashing technique should generate the same unique identifier even though some forms of attacks have been done to the original audio clip, given that the altered audio clip is reasonably similar (i.e., perceptually) to a human listener when comparing with the original audio clip. However, if the modified audio clip is audibly different or the attacks cause irritation to the listeners, the hashing technique should recognize such degree of changes and produce a different hash value from the original audio clip.

Content Categorization

Like anti-piracy, semantic categorizing of the audio content of audio clips often requires subjective comparisons to other existing audio works. Works of a similar nature are grouped into the same category. The content of audio clips may be semantically classified into any number of categories, such as classical music, conversation, hard rock, easy listening, polka, lecture, country, and the other such semantic categories.

Typically, such semantic categorization is subjectively determined by manual (i.e., human) subjective analysis of a work so that it may be grouped with an existing category. No such technique exists for automatically (i.e., without substantial human involvement) analyzing and categorizing the semantic audio content of audio clips.

SUMMARY

Described herein is a technology for recognizing the audio content of digital signals. The technology determines one or more hash values for the original audio content of a digital signal.

In one implementation described herein, audio content recognition facilitates identification of the original audio content and other audio content that is substantially perceptually same as the original audio content. An identification hash value uniquely identifies the original audio content to determine if the content of another digital signal is substantially perceptually same.

In another implementation described herein, audio content recognition facilitates semantic categorization of such original content so that it may be

grouped with other audio works in the same semantic category. A categorization hash values is used to group the original content with other audio works having similar categorization hash values; thereby, semantically classifying audio works having similar hash values as a semantic category of similar audio works.

This summary itself is not intended to limit the scope of this patent. For a better understanding of the present invention, please see the following detailed description and appending claims, taken in conjunction with the accompanying drawings. The scope of the present invention is pointed out in the appending claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like elements and features.

Fig. 1 is a schematic block diagram showing an embodiment of an implementation of the present invention claimed herein.

Figs. 2A-2C illustrate an implementation of the MCLT transform in accordance with an implementation of the present invention claimed herein.

Figs. 3A-1 and 3A-1 illustrate the time-frequency representation and the significance map, respectively, of given audio clip.

Figs. 3B-1 and 3B-1 illustrate the time-frequency representation and the significance map, respectively, of another audio clip.

Fig. 4 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Figs. 5A-5C illustrate some of the tasks performed by a statistic estimator in accordance with an implementation of the present invention claimed herein.

Fig. 6 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 7 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 8 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 9 is an example of a computing operating environment capable of implementing an implementation of the present invention claimed herein.

DETAILED DESCRIPTION

The following description sets forth one or more specific embodiments of an recognizer of audio-content in digital signals that incorporate elements recited in the appended claims. These embodiments are described with specificity in order to meet statutory written description, enablement, and best-mode requirements. However, the description itself is not intended to limit the scope of this patent.

Described herein are one or more exemplary implementations of a method and system of fusing portions of a print medium. The inventors intend these exemplary implementations to be examples. The inventors do not intend these exemplary implementations to limit the scope of the claimed present invention. Rather, the inventors have contemplated that the claimed present invention might also be embodied and implemented in other ways, in conjunction with other present or future technologies.

An exemplary embodiment of an recognizer of audio-content in digital signals may be referred to as an "exemplary audio recognizer."

Incorporation by Reference

The following co-pending patent applications are incorporated by reference herein (which are all assigned to the Microsoft Corporation):

- U.S. Patent Application Serial No. 09/390271, entitled "A Technique for Watermarking an Image and a Resulting Watermarked Image" filed Sept. 7, 1999;
- U.S. Patent Application Serial No. 09/390272, entitled "A Technique for Detecting a Watermark in a Marked Image" filed on Sept. 7, 1999;
- U.S. Patent Application Serial No. 09/316,899, entitled "Audio Watermarking with Dual Watermarks" filed on May 22, 1999;
- U.S. Patent Application Serial No. 09/614,660, entitled "Improved Stealthy Audio Watermarking" filed on July 12, 2000;
- U.S. Patent Application Serial No. _____, entitled "Robust Recognizer of Perceptually Similar Content" filed on April 24, 2001;
- U.S. Patent Application Serial No. _____, entitled "Derivation and Quantization of Robust Non-Local Characteristics for Blind Watermarking" filed on April 24, 2001;
- U.S. Patent Application Serial No. 09/259,669, entitled "A System and Method for Producing Modulated Complex Lapped Transforms" filed on February 26, 1999; and
- U.S. Patent Application Serial No. 09/421,986, entitled "System and Method for Hashing Digital Images" filed on October 19, 1999.

The following U.S. Patent is incorporated by reference herein: U.S. Patent No. 6,029,126, entitled "Scalable Audio Coder and Decoder" issued on February 22, 2000, and assigned to the Microsoft Corporation.

Introduction

Implementations, described herein, of the exemplary audio recognizer may be implemented (whole or in part) by an audio-content recognition system 100 and/or by a computing environment like that shown in Fig. 9.

An exemplary implementation is described herein as a technique for generally recognizing audio content of digital audio signals by hashing such signals to generate one or more hash values for each signal.

An exemplary implementation is described herein as a technique for identifying audio content of such signals by comparing identification hash values of signals. This exemplary implementation generates the same unique identifier (e.g., hash value) even though some forms of attacks have been done to the original digital audio signal, given that the altered signal is perceptually same to a human listener when comparing the altered signal with the original signal. However, if the altered signal is perceptually audibly different or the attacks cause irritation to the listeners, the hashing technique recognizes such degree of changes and produces a different hash value from the original signal.

An exemplary implementation is described herein as a technique for categorizing audio content of such signals by comparing categorization hash values of signals.

An exemplary implementation described herein is a technique that may be combined with techniques described in documents which are incorporated herein by reference.

Exemplary Applications

Implementations, described herein, of the exemplary audio recognizer are suitable for numerous applications, including the following (which are provided as examples and not as limitations): identification, searching & sorting in a database, semantic content categorization, and anti-piracy applications (such as watermarking). Some of the exemplary applications are discussed below:

Locating content in a database. Identification hash values may be stored and associated with specific audio content of a digital audio signal. When searching for such content, a search engine may look for a given hash value to locate the content. This is much more efficient that conventional techniques for searching for audio content in a database.

Semantic content categorization. This includes approximate matching of content between two digital audio signals. The hashing techniques of the exemplary embodiments have an intermediate hash value, which can be used to compare if two given items are similar. This hash value may also be called a categorization hash value.

This categorization hash value may be used to semantically classify audio content of audio works. Works of a similar nature tend to have categorization hash values that cluster together. Thus, these values are proximally near each other. This proximal range may be subjectively determined. For some semantic categories, the range may be large and for others it may be small.

The categorization hash can be computed incrementally. This means that if the exemplary embodiment may slide the window of the audio clip, one may compute the hash value on the new window from the old window without doing substantial reworking on the part that is common to old and new windows.

Anti-piracy search efforts. One may search for the pirated content of audio signals on the web, which might have been subjected to watermarking and/or malicious attacks, using a web crawler and a database of signal hash values.

Content dependent key generation (for watermarks): Since the watermarking technique must use secret keys, using the same key for millions of pieces of content may compromise the key. Thus, an exemplary embodiment may generate a hash value for an audio signal that may function as signal dependent keys. For an attacker without the knowledge of the secret key used, the hash value of a given content will be unpredictable.

Hashing

The use of hashing techniques, which map long inputs into short random-looking (i.e., uniformly distributed) outputs, are many and indeed wide-ranging: compilers, checksums, searching and sorting techniques, cryptographic message authentication, one-way hashing techniques for digital signatures, time stamping, etc. These techniques usually accept binary strings as inputs and produce a fixed length ("L") hash value. These techniques use some form of random seeds (i.e., keys).

The hash values produced by such techniques are viewed as useful because they typically have following desirable characteristics:

- Almost Uniformly Distributed —For any given input, the output hash value are uniformly distributed among the possible L-bit outputs.
- Approximate Pairwise Independent—For two distinct inputs, the corresponding outputs are statistically almost independent of each other.

The hashing technique implemented by various systems and methods, described herein, is denoted as H. Given an input signal I, the hashing technique H produces a short binary string X, as follows:

$$H(I) = X$$

The hashing technique H has the following properties:

- For any signal I_i, the hash of the signal, $H(I_i)$, is approximately uniformly distributed among binary strings of equal length.
- For two distinct signals, I_1 and I_2, the hash value of the first signal, $H(I_1)$, is approximately independent of the hash value of the second signal, $H(I_2)$, in that given $H(I_1)$, one cannot predict $H(I_2)$.
- If two signals I_1 and I_2 are perceptually the same or similar, the hash value of the first signal, $H(I_1)$, should equal the hash value of the second signal, $H(I_2)$.

The hashing techniques of the implementations, described herein, are similar in many respects to the hashing techniques described in some of the documents which are incorporated herein by reference. The hashing techniques, described herein, are particularly tailored to accommodate characteristics of the human auditory system and withstand various audio signal manipulation processes common to today's digital audio signal processing.

Perceptually Same and Perceptually Distinct

The exemplary audio recognizer treats two "perceptually same" audio signals as the same. Herein, a pair of digital audio signals are "perceptually same" when their identification hash values are the same (alternatively, substantially the same).

"Perceptually same" audio signals include those that sound as if they are they are substantially same to the human ear. As a first step in determining what it means to be perceptually same, one can use the standard Turing test used for formulating, for example, pseudo-randomness in cryptography. A listener is played two audio clips, one after another. Then the clips are played in random order and the listener should match up the clips. If the listener has no better than roughly a fifty-percent chance, the clips can be deemed perceptually same.

However, the use of the term is more general than that defined by the Turing test. Clips that have passed the Turing test may still be considered perceptually same when the listener is able to distinguish them only based on "slight and insubstantial differences." Regardless of such slight and insubstantial differences, the listener can clearly state that these clips "are the same audio clips for all practical purposes."

In contrast, a "perceptually distinct" digital goods is generally the converse of "perceptually same" digital goods. This may also be called "perceptually different" or "perceptually distinguishable".

Perceptually Similar

The exemplary audio recognizer treats two "perceptually similar" audio signals as different signals that should be categorized as similar. Herein, a pair of digital audio signals are "perceptually similar" when their categorization hash values are close in value (i.e., proximal). Signals that are "perceptually similar" may also be "perceptually same" if their identification hash values are the same.

Exemplary Hashing Technique of the Exemplary Implementations

The hashing technique of the exemplary implementations is an irreversible hashing function that operates on audio clips and yields a final hash value of a binary string of specified length. It also yields one or more intermediate has values. The hashing techniques of the exemplary implementations have the following criteria:

- The hash values are almost uniformly distributed with high probability.
- With high probability, the hash values for "perceptually audibly distinct" audio clips are different.
- With high probability, the hash values for "perceptually audibly same" audio clips are the same.

Assume X denotes a particular audio clip, X' denotes a modified version of this clip which is "perceptually same" as X, and Y denotes a "perceptually distinct" audio clip. Assume L is the final length of the hash and H(.) represents a hashing function. The following is a performance measure on the hash values that are produced:

$$D(H(X),H(Y)) \stackrel{\Delta}{=} \frac{\sum_{i=1}^{L} H(X_i) \otimes H(Y_i)}{L} \quad (1)$$

where $H(X_i)$ and $H(Y_i)$ are the values of H(X) and H(Y) respectively at the i^{th} location, and $\otimes$ stands for the *XOR* operation. Formula 1 is the ratio of the number of locations where H(X) and H(Y) are different to the total length of the hash.

The following are examples of a family of hashing functions, for use with the exemplary embodiments, with parameters: p, L (where $p \approx 1/2^L$).

- Randomization:

$$\Pr[H(X) = \alpha] = p \approx \frac{1}{2^L}, \forall \alpha \in \{0,1\}^L,$$

- Perceptually distinct audio clips X, Y:

$$\Pr[H(X) = \alpha \mid H(Y) = \beta] \approx \Pr[H(X) = \alpha], \forall \alpha, \beta \in \{0,1\}^L,$$

Thus, H(X) is not equal to H(Y), with overwhelming probability

- Perceptually same audio clips:

$$\Pr\left[H(X) = H(X')\right] \approx 1.$$

Therefore, apart from the randomization issue, the difference between identification hash values of audio clips may be represented as follows:

$$D(H(X), H(X')) = 0, D(H(X), H(Y)) > 0, \quad (2)$$

for all possible different ("perceptually distinct") audio clips X, Y and for all possible "perceptually same" audio clips X, X'.

Intermediate hash value. In the exemplary embodiments described herein, an intermediate hash value is calculated before the final (i.e., identification) hash value is calculated. The intermediate hash values are of length M , where M > L and have the following separation property:

$$D(H(X), H(X')) < 0.2, D(H(X), H(Y)) > 0.40, \quad (3)$$

In the exemplary embodiments, M has a value within range of 5L to 10L. To determine the intermediate hash value, decoding stages of first order Reed-Müller codes employed with a specialized pseudo-norm. Given the intermediate hash, the exemplary embodiment uses some generic techniques (e.g., list-decoding procedures) to generate a binary string with desired properties.

For more information on generating intermediate hash values, see U.S. Patent Application Serial No. _____, entitled "Robust Recognizer of Perceptually Similar Content" filed on April __, 2001.

Exemplary Audio Recognizer

Fig. 1 shows the audio-content recognition system 100, which is an example of an embodiment of the exemplary audio recognizer. The system 100 includes a transformer 110, a statistics estimator 120, an adaptive quantizer 130, and an error-correction decoder 140.

The transformer 110 obtains a digital audio signal 105 (such as an audio clip). It may obtain the signal from nearly any source, such as a storage device or over a network communications link. The transformer 110 puts the signal 105 in canonical form using a set of transformations. Specifically, the exemplary audio recognizer employs MCLT (Modulated Complex Lapped Transform) and obtain time-varying spectral characteristics, T_x 112, of the audio clip.

The statistics estimator 120 applies a randomized interval transformation in order to extract audible statistics, μ_x 122, of the signal. These audible statistics are expected to represent the audio signal in an irreversible manner while introducing robustness against attacks. For perceptually same audio clips, these statistics are likely to have close values (under suitable notions of metric) whereas for perceptually distinct audio clips they are far apart.

The adaptive quantizer 130 applies randomized rounding (i.e., quantization) to the outputs of the statistics estimator 120. The adaptive quantizer produces outputs, q_x 132, that are represented as binary vectors. Alternatively, the quantizer 130 may be non-adaptive.

The error-correction decoder 140 uses the decoding stages of an error correcting code to map similar values to the same point. The final output, h_x 142, is produced by the decoder 140. This final output is the final hash value. The

decoder also produces an intermediate hash value. Alternatively, error-correction decoder 140 may be a randomized Vector Quantizater (VQ) or list-decoder, or other similar devices.

The decoder 140 produces intermediate hash values such that the normalized Hamming distance between intermediate hash values of perceptually distinct audio signals is greater than 0.40 and the normalized Hamming distance between intermediate hash values of perceptually similar audio signals is less than 0.20. Of course, these ranges are provided as examples only and not for limitation.

Each of aforementioned components of the audio-content recognition system 100 of Fig. 1 is explained in greater detail below.

The Transformer 110 and MCLT

MCLT is a complex extension of MLT (Modulated Lapped Transform). MLT was introduced in and is used in many audio-processing applications, such as DolbyAC-3, MPEG-2. MCLT basis functions are found in pairs to produce real and complex parts separately. These basis functions are derived from MLT and they are phase-shifted versions of each other. It can be shown that they possess some intuitively pleasing properties such as perfect reconstruction and approximate shift invariance. The MCLT is a special case of 2x oversampled DFT (Discrete Fourier Transform) filter bank.

The MCLT is discussed more fully in U.S. Patent Application Serial No. 09/259,669, entitled "A System and Method for Producing Modulated Complex Lapped Transforms," which is incorporated by reference herein.

Figs. 2A-2C illustrate the MCLT scheme, like the one employed by the transformer 110 of the audio-content recognition system 100 of Fig. 1. Fig. 2A shows a timeline 210 of an input sequence of an audio signal (which is not shown). The sequence is broken into overlapping "blocks" (blocks 212-218) along the timeline, so that neighboring blocks intersect by half.

As shown in Fig. 2B, the MCLT transform is applied, by MCLT transformer 230, applied independently to each block (such as block "i") to produce spectral decomposition of size M. Assuming that the analysis and synthesis filters are of length 2M, then the number of the frequency bands is M in the spectral domain.

Fig. 2C represents the combination of the spectral decomposition of the blocks in order to form the time-frequency decomposition, T_x. The MCLT transform of the blocks are combined into a matrix to obtain the time-frequency representation of an audio clip. For Fig. 2C, Let M be the number of frequency bands and N be the number of the blocks and T_x is the MCLT matrix and $T_x(i, j)$ is the MCLT value at location (i, j) where $j = 0, 1, \ldots, N-1$.

MCLT can be used to define a "hearing threshold matrix" H_x, which is the same size as T_x, such that if $T_x(i, j) >= H_x(i, j)$, then $T_x(i, j)$ is audible.

Randomized Interval Transformation (Statistics Estimation)

The following is the definition of the significance map S_x: $S_x(i, j) = 1$, if $T_x(i, j) >= H_x(i, j)$; otherwise $S_x(i, j) = 0$; where $i = 0, 1,\ldots,M-1$ and $j = 0, 1,\ldots, N-1$.

Figs. 3A-1 and 3A-2 illustrate the time-frequency representation and corresponding significance map for an exemplary audio clip A. Figs. 3B-1 and

3B-2 illustrate the time-frequency representation and corresponding significance map for a different exemplary audio clip, clip B. Only the low-frequency portions of the time-frequency representations and the significance maps are depicted in these figures for the purposes of convenience.

As shown in Figs. 3A-1 and 3B-1, there is a striking pattern in time-frequency representation of the audio clips. Furthermore, this pattern has a slowly varying structure—with respect to both time and frequency. The exemplary audio recognizer captures this existing structure in a compact fashion via randomized interval transformations (i.e., "statistics estimation").

The statistics estimator 120 estimates statistics reflect characteristics of the signal 105. In order to achieve this purpose, the estimator 120 carries out estimation of statistics in the time-frequency plane. The estimator 120 exploits both local and global correlations. Correlations exist both along frequency axis and along the time axis. The estimator 120 may, for example, employ one or two exemplary approaches. Approach I operates along the frequency axis for each block (i.e., exploits correlations along the frequency axis for a given block). Approach II operates along the time axis for each frequency subband; and therefore, it exploits correlations along time.

Methodological Implementations of Approach I and Approach II:

Fig. 4 show methodological implementations of both Approach I and Approach II, which is performed by the estimator 120 of the audio-content recognition system 100 (or some portion thereof). These methodological implementations may be performed in software, hardware, or a combination thereof.

The methodology of Approach II is very similar to Approach I. The main difference is that in Approach I correlations along frequency are exploited instead of correlation along time while in Approach II correlations along time are exploited instead of correlation along frequency.

At 410 of Fig. 4, for each block, the estimator 120 determines if there exist sufficiently many entrees exceeding the hearing thresholds. If not pass to the next block, else collect the "significant" coefficients into a vector of size $M' \leq M$. More generally, the columns of T_x are used in Approach I and the rows of T_x are used in Approach II.

At 412, the estimator 120 performs a randomized interval transformation. Fig. 5A illustrates the concept of randomized "splitting" at a single level. At a single level of randomized splitting of a vector 510 (in either the frequency or time domain depending upon the Approach), first a randomization interval 520 (i.e., randomization region) is found that is to be symmetric around the midpoint 522. The ratio of the length of the randomization interval to the length of the whole vector 510 is the randomization tuning parameter for splitting. This may be a user-specified value.

Fig 5A shows that a random point 524 is chosen within this interval 520. This random point 524 is the point to do the splitting; as a result, two "chunks" are formed. Then this procedure is carried out a specified number of times—each time is a "level" of the splitting. The level is a function of M' and the expected number of coefficients desired within the smallest chunk. Hence, the expected number of coefficients that are within the smallest chunk is a user-specified parameter and determines the "level" for each block. As an example, in Fig. 5B, the recursive splitting is carried out twice, hence level = 2.

At 414 of Fig. 4, the first order statistics at each level for each chunk are estimated once the split points are randomly found. As an example, consider Fig. 5B. At second level of splitting, there are a total of four chunks (chunks 541, 542, 543, 544), and the estimator 120 collects the arithmetic means of these four chunks (alternatively, other statistical values can be collected, such as variance). Then the estimator 120 proceeds to the first level of splitting, and the estimator 120 computes the arithmetic means at that level—namely, arithmetic means of chunks 545 and 456 in Fig. 5B). Finally, at the zeroth level, the estimator 120 performs the arithmetic mean computation for the whole vector 510 (namely, chunk 547). All of these arithmetic means are collected in a statistics vector.

At 416, the process returns back to the beginning of the loop at 405 and repeat the steps of blocks 410-416 until all blocks have be processed. After all blocks have been processed, the loop ends at 418.

Fig. 5C illustrates the difference of Approach I 560 and Approach II 570 on the time-frequency representation 550 of an audio signal.

In Approach I, the estimator 120 estimates the statistics for each time block. With Approach I, the result of the estimator 120 is a statistics vector μ_f to denote the estimated means along the *frequency* axis.

In Approach II, the estimator 120 estimates the statistics for each frequency subband. With Approach II, the result of the estimator 120 is a statistics vector μ_t to denote the estimated means along the *time* axis.

Although both Approaches I and II of the exemplary embodiment are designed to estimate the first order statistics at this point, an alternative embodiment may include an estimation of any order statistics. In tests, it has been observed that first order statistics yield better results than second order statistics at

this point. Regardless of which approach is employed (Approach I or II), the function of the adaptive quantizer 130 and the error correction decoder 140 is the same.

Another exemplary embodiment employs a third approach, Approach III, which combines Approaches I and II. In this approach, statistics are estimated based on random rectangles in the time-frequency plane. The shape of these rectangles may be adjusted to capture the appropriate characteristics.

Input Adaptive Quantization

The adaptive quantizer 130 produces discrete level of outputs given the statistics vector as the input. Meanwhile, the adaptive quantizer 130 effectively increases robustness properties of the hashing techniques (employed by the exemplary audio recognizer) and augments the amount of randomization.

In signal processing, the traditional way of producing discrete level of outputs from continuous level of inputs is termed as "quantization." Assume Q is the number of quantization levels (i.e., the cardinality of the set of discrete level of outputs). Also assume $\mu(j)$ denote the j^{th} element of a given statistics vector μ and $\mu'(j)$ denote its quantized version. In conventional quantization schemes, the quantization rule is completely deterministic and given by

$$\Delta_i \le \mu(j) < \Delta_{i+1} \Rightarrow \mu'(j) = i, i = 0,1,...,Q-1,$$

where the interval $[\Delta_i, \Delta_{i+1})$ is termed as i^{th} quantization bin. With the description herein of the exemplary embodiment, the focus is on the location of quantization bins rather than the reconstruction levels. In traditional quantization schemes in

signal processing, the reconstruction levels are significant since quantization is usually applied as a part of a compression scheme. However, apart from the indexing issues, the reconstruction levels are not of significance for the exemplary embodiment described herein. Therefore, without loss of generality, the reconstruction levels are assumed to be given by integer indexes from the set $(0,1,\ldots,Q-1\}$.

Typically, the input statistics vector comes from a distribution that is highly biased at some points. To account for this "colored" nature of the statistics distribution, the exemplary embodiment employs an "adaptive quantization" scheme, which takes of possible arbitrary biases at different locations of the distribution of the statistics. In particular, the exemplary embodiment uses a normalized histogram of μ as the distribution of the statistics. A normalized histogram is usually very resistant against "slightly inaudible" attacks, thus such an adaptive scheme does not really bring a burden in terms of robustness. Furthermore, a normalized histogram approximates the p.d.f. (probability density function) of the marginal density functions of $\mu(j)$ are the same for all j, and the approximation error diminishes as the size of μ tends to in infinity. Thus, within the exemplary embodiment $\{\Delta_i\}$ is designed such that

$$\int_{\Delta_{i-1}}^{\Delta_i} p_\mu(t)dt = \frac{1}{Q}, i = 0,1,\ldots,Q-1.$$

where p_μ stands for the normalized histogram of the input statistic vector μ. The intervals $[\Delta_i\text{-}1, \Delta_i)$ determine the quantization bins. Furthermore, define the "central points", $\{C_i\}$, such that

$$\int_{\Delta_{i-1}}^{C_i} p_\mu(t)dt = \int_{C_i}^{\Delta_i} p_\mu(t)dt = \frac{1}{2Q}, i = 0,1,\ldots,Q-1.$$

Now around each Δ_i, we introduce a randomization interval $[L_i,U_i]$ such that

$$\int_{L_i}^{\Delta_i} p_\mu(t)dt = \int_{\Delta_i}^{U_i} p_\mu(t)dt, i = 0,1,...,Q-1,$$

In other words, the randomization interval is symmetric around Δ_i for all i and we also impose the constraint that $C_i \le L_i$ and $U_i \le C_{i+1}$. The exact location of these randomization intervals are determined by "Randomization Factor",

$$\text{Randomization Factor} \quad = \frac{\int_{L_i}^{\Delta_i} p_\mu(t)dt}{\int_{\Delta_{i-1}}^{\Delta_i} p_\mu(t)dt}, i = 0,1,...,Q-1$$

where "randomization factor" is a parameter that determines the amount of randomization at the output of quantization and is a user-specified number that can clearly take values in the range [0,½]. This the p.d.f.-adaptive randomized quantization rule of the exemplary embodiment:

$$L_i \le \mu(j) \le U_i \quad \Rightarrow \quad \mu'(j) = \begin{cases} i & \text{with probability } \dfrac{\int_{L_i}^{\mu(j)} p_\mu(t)dt}{\int_{L_i}^{U_i} p_\mu(t)dt} \\ i-1 & \text{with probability } \dfrac{\int_{\mu(j)}^{U_i} p_\mu(t)dt}{\int_{L_i}^{U_i} p_\mu(t)dt} \end{cases}$$

and

$$C_i \le \mu(j) \le L_i \quad \Rightarrow \quad \mu'(j) = i-1 \text{ with probability 1,}$$

$$U_i \le \mu(j) < C_{i+1} \quad \Rightarrow \quad \mu'(j) = i \text{ with probability 1}$$

Under such a randomized quantization rule, if $L_i \leq \mu(j) \leq U_i$, then $E[\mu'(j)] = \Delta_i$. This choice of the "Randomization Factor" offers a trade-off: As this factor increases, the amount of randomization at the output increases, which is a desired property; however, this also increases the chances of being vulnerable to attacks and modifications especially because in that case the security of the system relies, to a great extent, on keeping the randomization key as a secret. Thus, choosing a suitable range for "Randomization Factor" is a delicate issue. In the exemplary embodiment, a user-specified input parameter determines this issue. With such user-specified input parameter, the user may balance the degree of randomization desired against security concerns to achieve a customized result.

Error Correction Decoding

Once the exemplary embodiment has the quantized statistics, the next step is to convert these values into a binary bit stream (i.e., digital signal) and shorten the length of the overall stream in such a way that "perceptually same" audio clips are mapped to binary strings that are close to each other and "perceptually distinct" audio clips are mapped to binary strings that are far away from each other.

In order to achieve this purpose, the exemplary embodiment employs first order Reed-Müller codes at this point. Those who are skilled in the art understand and appreciate that other error correction schemes as well as possibly randomized Vector Quantization techniques may be employed and remain within the spirit and scope of the claimed invention.

Reed-Müller codes are a class of linear codes over GF(2) that are easy to describe and have an elegant structure. The generator matrix G for the first order Reed-Müller code of blocklength 2^m is defined as an array of blocks:

$$G = \begin{bmatrix} G_0 \\ G_1 \end{bmatrix}$$

where G_0 is a single row consisting of all ones and G_1 is a matrix of size m by 2^m. G_1 is formed in such a way that each binary m-tuple appears once as a column. Thus, the resulting generator matrix is of size m+1 by 2^m. More details on error correcting codes and Reed-Müller codes by found in *Theory and Practice of Error Control Codes* (1983) by R. Blahut.

Although there exist computationally efficient techniques for decoding with Reed-Müller codes (decoding with majority logic), the exemplary embodiment is using an exhaustive search on the input word space for the sake of simplicity. Unlike traditional decoding schemes that use Hamming distance as the error metric, the decoding scheme of the exemplary embodiment uses an error measure, which is termed as "Exponential Pseudo Norm" (EPN). It is more suitable than traditional error metrics (such as Hamming distance) for multimedia (image and audio) hashing problems.

Assume $\underline{x}_D$ and $\underline{y}_D$ are two vectors of length L such that each component of these vectors belongs to the set {0,1,...,Q-1} where $\log_2 Q$ is a positive integer. Similarly, assume x and y are binary representations of the vectors $\underline{x}_D$ and $\underline{y}_D$ respectively, where each decimal component is converted to the binary format by using $\log_2 Q$ bits. The lengths of $\underline{x}$ and $\underline{y}$ are therefore going to be both L $\log_2 Q$. EPN is defined between the binary vectors $\underline{x}$ and $\underline{y}$ as

$$EPN(x,y) \overset{\Delta}{=} \sum_{i=1}^{L} K^{|x_D(i)-y_D(i)|},$$

where $x_D(i)$ and $y_D(i)$ denote the ith elements of the vectors $\underline{x}_D$ and $\underline{y}_D$ respectively. EPN ($\underline{x}$, $\underline{y}$) is actually a function of Q and K as well; however, for the sake of having a clean notation the exemplary embodiment are embedding these values in the expression and simply assuming that these values are known within the context of the problem.

Q is the number of quantization levels, and K is the "exponential constant" that determines how EPN penalizes large distances more. The results are approximately insensitive to the value of K if it is chosen to be large enough. Since part of the aim of the exemplary embodiment is to clearly distinguish between close and far values in the decimal representations of binary strings, EPN is suitable for incorporations into the hashing techniques of the exemplary embodiment.

Examples of methodological acts performed by the error correction decoder 140 are as follows:

- Divide the quantized data into chunks of length that is user-specified.
- Convert them into binary format by using $\log_2 Q$ bits for each component, where Q is the number of quantization levels.
- Form the generator matrix of first order Reed-Müller code where the length of the codewords is as close as possible to the length of the binary representation of the chunks.

- For each possible input word (there are a total of 2^{m+1} possible input words for a generator matrix of size m+1 by 2^m), generate the corresponding output word.
- Find the EPN between each corresponding output word and the quantized data.
- Pick up the input word that yields the minimum amount of EPN.

Methodological Implementation of the Exemplary Audio-Content Recognizer

Fig. 4 shows an illustrative methodological implementation of the exemplary audio-content recognizer performed (wholly in part) by the audio-content recognition system 100 (or some portion thereof). This methodological implementation may be performed in software, hardware, or a combination thereof.

Audio-Content Identification Methodological Implementation

Fig. 6 illustrates an audio-content identification methodological implementation of the exemplary audio-content recognizer. At 610 of Fig. 6, the exemplary audio-content recognizer retrieves a subject audio clip from a database of audio clips or some other source of such audio signals. Once a subject clip is chosen, the exemplary audio-content recognizer, at 612, transforms it, in accordance with the transformation performed by the transformer 110, described above.

At 614 of Fig. 6, the exemplary audio-content recognizer estimates statistics of the transformed clip, in accordance with the estimation performed by

the statistics estimator 120, described above. At 616, the exemplary audio-content recognizer adaptively quantizes the estimated statistics of the transformed clip, in accordance with the quantization performed by the adaptive quantizer 130, described above. At 618, the exemplary audio-content recognizer performs error-correction decoding on the results of the adaptive quantization, in accordance with the decoding performed by the decoder 140, described above.

At 620 of Fig. 6, it determines the hash values based upon the result of the above steps. The hash values include an intermediate hash value (i.e., categorization hash value) and a final hash value. [These hash values are recognition representations of the audio content of the original audio clip. That is because these hash values may be used to recognize (and even identify) the audio content within an audio clip.

At 622 of Fig. 6, the resulting hash values are displayed and stored. These values are stored in a database in association with the original subject clip from which the values were calculated.

Piracy Detection Methodological Implementation

Fig. 7 illustrates a piracy detection methodological implementation of the exemplary audio-content recognizer. At 756 of Fig. 7, the exemplary audio-content recognizer retrieves a hash value of a selected audio clip. More particularly, it retrieves the final hash value (i.e., identification hash value) of such clip from a database of audio clips or some other source of such clips.

Fig. 7 also shows the audio-content identification method of Fig. 6 at block 752. The method 752 calculates a final hash value of an audio clip 750 that is suspected of being a copy of the selected clip retrieved by block 756. At 754, the

exemplary audio-content recognizer retrieves the calculated final hash value of the suspect audio clip 750 from the audio-content identification method of block 752. Of course, this can be reversed so that the method 752 provides the hash value of the *selected* clip while block 752 provides the hash value of the *suspected* clip.

At 758, the exemplary audio-content recognizer compares the hash values of the two clips (suspect clip 750 and selected clip of 756) to determine if they substantially match. Substantially matching means that the two hash values are close enough in value to reasonably conclude that the two clips have the same hash values within a margin of error.

If the result of such comparison is no substantial match, then the exemplary audio-content recognizer indicates, at 760, that the suspect clip 750 is not a substantial copy of the selected clip of 756. In other words, no piracy is detected if the final hash values of compared clips do not substantially match. At 764, this process ends.

However, if the result of such comparison is a substantial match, then the exemplary audio-content recognizer indicates, at 762, that the suspect clip 750 is a substantial copy of the selected clip of 756. In other words, piracy is detected if the final hash values of compared clips substantially match. At 764, this process ends.

Audio Content Categorization Methodological Implementation

Fig. 8 illustrates an audio content categorization methodological implementation of the exemplary audio-content recognizer. At 816 of Fig. 8, the exemplary audio-content recognizer retrieves a hash value of a selected audio clip.

More particularly, it retrieves the intermediate (i.e., categorization) hash value of such clip from a database of audio clips or some other source of such clips.

In dashed box 805, Fig. 8 also shows an alternative way of getting an intermediate hash value of the selected clip. This is by processing the clip using the audio-content identification method of Fig. 6 at block 812. The method 812 calculates an intermediate (i.e., categorization) hash value of the selected clip. At 814, the exemplary audio-content recognizer retrieves the calculated intermediate hash value of the selected audio content-base clip 810 from the audio-content identification method of block 812.

At 820, the exemplary audio-content recognizer uses the intermediate hash value of the selected clip to group such clip with others of similar (i.e., proximal) intermediate hash values. In other words, based upon the intermediate hash value of a given clip, the exemplary audio-content recognizer groups the given clip with other clips having similar intermediate hash values. Thus, the hash values of all clips in a given grouping are clustered together (i.e., proximal each other). Although these groupings are somewhat objectively determined, the subjective nature of the content of clips within a grouping will be similar to that of the content of others within the grouping.

The exemplary audio-content recognizer uses the categorization hash value of the selected clip to group such clip with others of similar (i.e., proximal) categorization hash values. In other words, based upon the categorization hash value of a given clip, the exemplary audio-content recognizer groups the given work with other works having similar categorization hash values. Thus, the hash values of all works in each grouping are clustered together (i.e., proximal each other). Although these groupings are primarily objectively determined, the

subjective nature of the content of works within a grouping will be similar to that of the content of others within the grouping.

The boundaries between groupings are determined manually or automatically. Manually, a person selects the boundary between groupings using the natural clustering seen after many clips have been categorized. Automatically, a system mathematically selects the boundary between groupings to be some point between (perhaps halfway) the centers of the groupings. Of course, other such techniques may to used to determine boundaries. These techniques may be fully automatic, fully manual, or some combination

At 822, the exemplary audio-content recognizer stores the categorization results in a database. At 824, the process ends.

Exemplary Computing System and Environment

Fig. 9 illustrates an example of a suitable computing environment 900 within which an exemplary audio recognizer, as described herein, may be implemented (either fully or partially). The computing environment 900 may be utilized in the computer and network architectures described herein.

The exemplary computing environment 900 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computing environment 900 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary computing environment 900.

The exemplary audio recognizer may be implemented with numerous other general purpose or special purpose computing system environments or

configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, thin clients, thick clients, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

Exemplary audio recognizer may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Exemplary audio recognizer may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

The computing environment 900 includes a general-purpose computing device in the form of a computer 902. The components of computer 902 can include, by are not limited to, one or more processors or processing units 904, a system memory 906, and a system bus 908 that couples various system components including the processor 904 to the system memory 906.

The system bus 908 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry

Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

Computer 902 typically includes a variety of computer readable media. Such media can be any available media that is accessible by computer 902 and includes both volatile and non-volatile media, removable and non-removable media.

The system memory 906 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 910, and/or non-volatile memory, such as read only memory (ROM) 912. A basic input/output system (BIOS) 914, containing the basic routines that help to transfer information between elements within computer 902, such as during start-up, is stored in ROM 912. RAM 910 typically contains data and/or program modules that are immediately accessible to and/or presently operated on by the processing unit 904.

Computer 902 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example, Fig. 9 illustrates a hard disk drive 916 for reading from and writing to a non-removable, non-volatile magnetic media (not shown), a magnetic disk drive 918 for reading from and writing to a removable, non-volatile magnetic disk 920 (e.g., a "floppy disk"), and an optical disk drive 922 for reading from and/or writing to a removable, non-volatile optical disk 924 such as a CD-ROM, DVD-ROM, or other optical media. The hard disk drive 916, magnetic disk drive 918, and optical disk drive 922 are each connected to the system bus 908 by one or more data media interfaces 926. Alternatively, the hard disk drive 916, magnetic disk drive 918,

and optical disk drive 922 can be connected to the system bus 908 by one or more interfaces (not shown).

The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 902. Although the example illustrates a hard disk 916, a removable magnetic disk 920, and a removable optical disk 924, it is to be appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the exemplary computing system and environment.

Any number of program modules can be stored on the hard disk 916, magnetic disk 920, optical disk 924, ROM 912, and/or RAM 910, including by way of example, an operating system 926, one or more application programs 928, other program modules 930, and program data 932. Each of such operating system 926, one or more application programs 928, other program modules 930, and program data 932 (or some combination thereof) may include an embodiment of a digital audio signal hashing unit, a watermark encoder, transformer, a statistics estimator, an adaptive quantizer, an error-correction decoder, and a hasher.

A user can enter commands and information into computer 902 via input devices such as a keyboard 934 and a pointing device 936 (e.g., a "mouse"). Other input devices 938 (not shown specifically) may include a microphone,

joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to the processing unit 904 via input/output interfaces 940 that are coupled to the system bus 908, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

A monitor 942 or other type of display device can also be connected to the system bus 908 via an interface, such as a video adapter 944. In addition to the monitor 942, other output peripheral devices can include components such as speakers (not shown) and a printer 946 which can be connected to computer 902 via the input/output interfaces 940.

Computer 902 can operate in a networked environment using logical connections to one or more remote computers, such as a remote computing device 948. By way of example, the remote computing device 948 can be a personal computer, portable computer, a server, a router, a network computer, a peer device or other common network node, and the like. The remote computing device 948 is illustrated as a portable computer that can include many or all of the elements and features described herein relative to computer 902.

Logical connections between computer 902 and the remote computer 948 are depicted as a local area network (LAN) 950 and a general wide area network (WAN) 952. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When implemented in a LAN networking environment, the computer 902 is connected to a local network 950 via a network interface or adapter 954. When implemented in a WAN networking environment, the computer 902 typically includes a modem 956 or other means for establishing communications over the

wide network 952. The modem 956, which can be internal or external to computer 902, can be connected to the system bus 908 via the input/output interfaces 940 or other appropriate mechanisms. It is to be appreciated that the illustrated network connections are exemplary and that other means of establishing communication link(s) between the computers 902 and 948 can be employed.

In a networked environment, such as that illustrated with computing environment 900, program modules depicted relative to the computer 902, or portions thereof, may be stored in a remote memory storage device. By way of example, remote application programs 958 reside on a memory device of remote computer 948. For purposes of illustration, application programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computing device 902, and are executed by the data processor(s) of the computer.

Computer-Executable Instructions

An implementation of an exemplary audio recognizer may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

Exemplary Operating Environment

Fig. 9 illustrates an example of a suitable operating environment 900 in which an exemplary audio recognizer may be implemented. Specifically, the exemplary audio recognizer(s) described herein may be implemented (wholly or in part) by any program modules 928-930 and/or operating system 928 in Fig. 9 or a portion thereof.

The operating environment is only an example of a suitable operating environment and is not intended to suggest any limitation as to the scope or use of functionality of the exemplary audio recognizer(s) described herein. Other well known computing systems, environments, and/or configurations that are suitable for use include, but are not limited to, personal computers (PCs), server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, wireless phones and equipments, general- and special-purpose appliances, application-specific integrated circuits (ASICs), network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

Computer Readable Media

An implementation of an exemplary audio recognizer may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise "computer storage media" and "communications media."

"Computer storage media" include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

"Communication media" typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media.

The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

Conclusion

Although the invention has been described in language specific towards digital audio signals, it is to be understood that the invention defined in the appended claims is not necessarily limited to digital audio signals. Rather, it may

apply to other digital signals (e.g., images, multimedia, video, film, data, information, text, etc.)

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.